Method and arrangement for processing a signal using a digital processor having a given word length.

FIELD OF THE INVENTION

The invention relates to methods and arrangements for processing a signal using a digital processor having a given word length.

BACKGROUND OF THE INVENTION

It is well known that common digital signal processing (DSP) operations like Fast Fourier Transforms (FFT), convolutions, Discrete Cosine Transforms (DCT), etc., often require large dynamical ranges for the variables employed in such algorithms. This leads to implementations using floating point arithmetic rather than with fixed point arithmetic, because the latter yield larger rounding noise at equal word-length. Let us first recall the distinction between integer or fixed-point arithmetic on the one hand and floating point arithmetic on the other.

Our goal is to represent numbers in the memory or registers of a computer or digital circuit in the form of binary digits ('0's and '1's). Because of their discrete nature we can only represent a finite set of numbers, all other numbers are "rounded" or "truncated" to one of the representable values, leading to quantization noise. For the sake of the argument let us focus on numbers between –1.0 and 1.0, and say that we have 16-bits available to represent numbers in this range.

– for fixed point numbers, all representable numbers are of the form: $n \cdot 2^{-15}$, where n is an integer in the range $[-2^{15} \ldots 2^{15}-1]$. The representable numbers are uniformly spaced. The dynamic range, which is the ratio of the largest to the smallest representable number is $2^{15} \approx 10^{5}$.

– for floating point numbers, all representable numbers are of the form $s \cdot (0.5+m/2^{a+1}) \times 2^{e-b}$, where m is an a-bit integer, and $(0.5+m/2^{a})$ is called the "mantissa" and obviously lies between 0.5 and 1. s is a 1-bit "sign", and e is called the "exponent", a (15-a)-bit number, and b is the exponent-bias. As an example take a 7-bit mantissa and an 8-bit exponent. Then the range 0.5…1 (set e=b) contains 128 representable numbers 1/256 apart. The range 0.25…1 (set e==b-1) also contains 128 representable number 1/512 apart., etc. We see that the representable numbers are packed closer together, the closer we get to 0, in a logarithmic fashion. The exponent bias *b* sets the origin of this quasi-logarithmic scale. In this example the dynamic range is about $2^{256} \approx 10^{77}$.

Floating point numbers are a trade-off between a large dynamic range , and locally uniform distribution of representable numbers. This meshes nicely with the idea that in many relevant computations we need to represent small numbers with a small granularity and large numbers with a large granularity. Another way to say this: the floating point representation matches the "natural distribution of numbers", which is roughly logarithmic, much more closely. For that reason, in practice floating point calculations almost invariably lead to much more accurate results than fixed point calculations with words of the same size (number of bits).

The major drawback of floating point numbers is that they require more complex hardware to perform additions, multiplication etc. E.g. for a floating point addition, both operands have to be normalized to the same exponent, followed by an ordinary addition, and a final re-scaling of the exponent. In software floating-point operations are therefore usually much slower.

In the case of watermark detection, DSP operations like FFTs must happen accurately: a watermark is carefully hidden in the content (often in the least significant bits) and so the signal processor must proceed with care so as not to lose it. However for watermarking in copy-protection or tracing applications, cost is a major issue: it is not a feature which can warrant a higher price in the store. A manufacturer of watermark detectors has two choices to control the accuracy:

- use a floating point implementation with relatively high hardware cost (or high CPU-load for software)
- use a fixed point implementation, but considering the statements about accuracy above, one is forced to use much longer words for an integer implementation than for a floating points. This also drives up the cost if many memory words are needed for storage, and consequently a lot of memory bandwidth is needed.

Applicant's International Patent Application WO 99/45707 discloses a watermark embedding system (hereinafter referred to as JAWS) to which the invention is particularly applicable. A watermark, which is typically a pseudo random noise sequence or pattern, is added to a motion video signal in the spatial signal domain. For complexity reasons, the same watermark is embedded in every image (field or frame) of the video signal.

To reduce the complexity even more, a small watermark pattern is tiled over the image. A typical tile size is 128×128 pixels.

As in many watermark schemes, the watermark detection method is based on correlating the suspect signal with the pseudo random noise sequence. If the correlation exceeds a given threshold, the watermark is said to be present. In the JAWS watermark detector, the tiles of a number of images are folded into a 128×128 buffer. Detection is then performed by correlating the buffer contents with the small watermark pattern.

Fig. 1 shows the signal processing steps of the watermark detection process. The contents of the 128×128 fold buffer **11** is applied to a two-dimensional FFT **12**. In a Symmetrical Phase Only Matched Filtering (SPOMF) step **13**, the phase of the frequency coefficients is extracted and subsequently correlated **(14)** with a frequency domain representation of the watermark **15** to be detected. An inverse FFT operation **16** on the results of this correlation process yields a 2-dimensional array of correlation values. If a significant peak is found in this array **(17a)**, the contents is considered to be watermarked. Optionally and advantageously, the contents is also watermarked with a spatially shifted version of the same watermark. In that case, a further peak is searched for **(17b)**. Their relative positions represent an embedded payload, which is decoded by a payload decoder **18**.

Since for algorithms like JAWS, memory is the largest cost-factor, a floating point implementation with 17-bit words (8-bit mantissa, 8-bit exponent) was initially developed. For instance, in the 2D FFT-step **12**, if one wanted to use integers, one would need about 20...24 bits (depending on the video-content) to get similar accuracy to the 17-bit floating-point implementation.

From the literature there are many methods known which can help to reduce the word-length for integer FFTs e.g.:

– insertion of guard-bits (shifting the input to the right by k bits, if the signal processing step is expected to increase the dynamic range by k-bits). An FFT on $N=2^n$ points increases the dynamic range by N (worst case), so the required insertion of n guard bits is like pre-dividing the input by a factor of N.
– using block floating points. Block floating points are really (e.g. 16-bit) fixed points which represent a different range of numbers (like [-1...1], or [-¼ ... ¼] or [-8...8]) at different stages in the processing step, depending on the required dynamic range. For instance in an FFT one would choose a new range for the 16-bit variables to represent, after every one of the n stages.

Although these methods are helpful, in general they still cause too much quantization noise to allow e.g. robust watermark detection.

OBJECT AND SUMMARY OF THE INVENTION

It is one of the purposes of this invention disclosure to show how it is possible to use fixed-point implementation to do signal processing with words which are no longer than the floating point implementation.

In accordance with the invention, the signal is pre-processed by a pre-processor which reduces the word length and which is invariant with respect to the subsequent process. The expression "being invariant" means that if the pre-processor operated with infinite accuracy, it would have no effect on the subsequent process.

With the invention is achieved that if such a pre-processor operates with finite accuracy, it will reduce the quantization noise. Advantageous embodiments are defined in the appended sub-claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows a block diagram indicating the signal processing steps involved to perform watermark detection using the JAWS system.

Fig. 2 shows pictures illustrating the operation of the JAWS watermark detection system which is shown in Fig. 1.

Fig. 3 shows a block diagram of the JAWS watermark detection architecture including a pre-filter in accordance with the invention.

Fig. 4 shows the power spectrum densities of the outputs of the two-dimensional FFTs without (left) and with (right) pre-filtering in accordance with the invention.

Fig. 5 shows a diagram illustrating watermark detection reliability without and with pre-filtering in accordance with the invention.

Fig. 6 shows a block diagram illustrating an algorithm for performing cyclical correlations using pre-filters to compress the dynamic range of the input and a post-filter to undo the effects of the pre-filter.

DESCRIPTION OF EMBODIMENTS

For many applications the statistics of the input-signal to the signal processing step is well known. For instance in the case of the watermark detector shown in Fig. 1, the video-content is strongly peaked in the low horizontal and vertical frequencies. This is illustrated in Fig. 2, where the left picture shows a typical contents of the folding buffer **11** (here 1 second of folded video), and the right picture shows an intensity plot of the spectrum at the output of the FFT **12**, with a dynamic range of about 21 bits. The 0-frequency is at the center of the picture. The horizontal and vertical DC-frequencies cause FFT **12** to overflow. Preventing overflow by re-scaling yields even more quantization noise. This quantization noise ultimately obscures the watermark.

According to this invention it is preferred to first apply a (high-pass) filter to the input of the FFT which suppresses the low frequencies with large amplitude, thus reducing the required dynamic range. In fact the filter should be chosen such that it emphasizes those frequencies that contain most of the watermark energy, and cause least quantization noise in that energy range.

Fig. 3 shows a block diagram of the JAWS watermark detection architecture in accordance with the invention. The diagram is the same as in Fig. 1, but now includes a pre-filter **19** to reduce dynamic-range / overflow / quantization noise problems in the two-dimensional FFT **12**.

As an example, the contents of fold buffer **11** (cf. the left picture in Fig. 2) is filtered with the two-dimensional filter:

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

before inputting it into the 2D FFT. The result thereof is shown in Fig. 4. In this Figure, the left diagram shows the two-dimensional power density spectrum of the output of FFT **12** without pre-filtering. The vertical axis is in powers of 2. It is clear that about 21 bits of dynamic range are required to represent the bulk of the coefficients. The right diagram shows the output of the FFT with pre-filtering, using the filter mentioned above. The required dynamic range has been reduced to about 8-bits. The 2D spectra have been projected onto the plane given by x+y=0.

It is clear that the pre-filtering step causes a major decrease in required dynamic range to represent the (intermediate) result. In fact in combination with use of 16-bit block-floating point variables throughout the rest of the algorithm, the peak-reliability is almost indistinguishable from a 32-bit floating point implementation. Fig. 5 shows the watermark detection reliability as a function of time for a video sequence, processed once with a 32-bit floating point detector (solid line **51**) and once with a 16-bit integer detector (dashed line **52**).

Because of the normalization step (SPOMF's phase-extraction **13** right after the 2D FFT **12**), the effect of the input filter is purged. In other words: although with infinite precision variables the pre-filter has no effect (its effects are removed by SPOMF), it does reduce quantization noise significantly for a fixed-point implementation.

The advantages of pre-filtering are thus:

– cheaper integer-only hardware / smaller CPU-load for integer-only software
– allows implementation on integer-only DSP. The vast bulk of DSP in use today, and certainly the only DSPs available in the low-end market, are integer processors. Floating-point DSPs are very expensive.
– reduces memory consumption (for JAWS by a factor 16 to 18)
– reduces memory bandwidth requirements significantly. Some implementors will not embed the 72 kbytes of RAM needed by the JAWS system on board of a chip but rather re-use 72Kbytes of an external DRAM memory chip to (like from a 512 kb drive-buffer in a DVD-ROM drive). Since the memory data bus usually has a width of 16-bit, or 32 bits, an 18-bit transfer requires about (32/18) times more data to move across the bus than necessary. This is particularly problematic in high-speed (16×) DVD-ROM drives, where memory bandwidth is at a premium.

Although the invention emerged out of research in the field of JAWS watermarking, it will be appreciated that the method is general enough to benefit other watermarking methods, or indeed signal processing in general. By way of example, Fig. 6 shows an arrangement for computing the (cyclical) correlation between 2 patterns *A* (**61**) and *B* (**62**) by performing a multiplication (**67**) in the frequency domain. In accordance with the invention, appropriately chosen pre-filters *F* (**63**) and *G* (**64**) are employed, which are adapted to the statistical behavior of the respective patterns *A* and *B* so as to control quantization noise in the respective subsequent FFTs **65** and **66**. The effect of the filter (which scales the frequency components) is undone by a post-filter **68** before the inverse FFT **69** is applied. If, as often is the case, pattern *B* is fixed, the post-filtering step may be combined with the pre-filtering and FFT of pattern *B*.

As another example, consider the fingerprinting technique described in the paper "Robust Audio Hashing for Content Identification," by Jaap Haitsma, Ton Kalker and Job Oostveen, presented at the Content-Based Multimedia Indexing conference 2001, Brescia, Italy. In this technique a "fingerprint" of an audio signal is generated by splitting its power spectrum in a series of frequency bands and coding differences between these bands, both with respect to frequency and with respect to time, in a small number of bits. The fingerprint thus obtained is robust against a wide range of signal distortions, such as MP3 compression, noise addition, all-pass filtering, etc. Typically, the frequency bands considered cover the interval from 300 Hz to 3 kHz.

The power spectrum is obtained by applying a FFT to the downsampled and windowed input signal. As long as a floating-point algorithm is used this is just fine. However, quite often the power spectrum contains a peak near DC, which is substantially higher than the values in the frequency range of interest. This results in excessive quantization noise in that frequency range if an integer FFT with small dynamic range is used. Evidently, this may readily lead to spurious bit errors in the fingerprint, not caused by actual signal distortions, but caused by a deficiency of the implementation. The solution is to remove the DC peak by applying a high-pass filter to the input signal prior to performing the FFT, or alternatively, to apply a band-pass filter which only selects the frequencies of interest.

The invention can be summarized in the following manner. A digital signal processor operating with integer arithmetic circuits has a certain accuracy. Each processing step (multiplication, addition) increases the number of bits (the word length). For example, the Fast Fourier Transform having a butterfly structure requires a plurality of such processing steps to be performed. In practical implementations, the processing steps are recursively performed by a single integer arithmetic circuit having a given word length, say N. After each step, the word length of the signal is reduced to the given word length N by rounding, truncation, or some other smart form of quantization. An obvious way to prevent quantization errors is to scale down the input signal. However, this results in quantization errors to be already introduced in the input signal. For processes such as watermark detection this is fatal, since the least significant bits of the input signal constitute precisely the place where the watermark is embedded.

In accordance with the invention, the signal is pre-processed by a pre-processor which reduces the word length and which is invariant with respect to the subsequent process. The expression “being invariant” means that if the pre-processor operated with infinite accuracy, it would have no effect on the subsequent process. If such a pre-processor operates with finite accuracy, it will reduce the quantization noise. The high-pass pre-filter described above with reference to the JAWS watermark detection process fulfills this condition, because it is a zero-phase filter and the watermark to be detected is carried by the phase of Fourier coefficients.